Создание 3D пещеры в 3ds Max

Автор: Д.Ю. Береженко

Конечную обработку модели пещеры произвожу в ПО 3dsMax от компании Autodesk. Основная причина выбора данной САПР (с натяжкой можно отнести 3dsMax к этому типу ПО) – это личная привязанность к серии данного программного продукта. Имеется множество хороших САПР, но 3dsMax максимально нагляден и прост в использовании.

Существует несколько способов импорта модели пещеры в 3dsMax. Одно из распространённых решений импорт пещеры в формате DFX (генерируется программой Thereon, Compass).

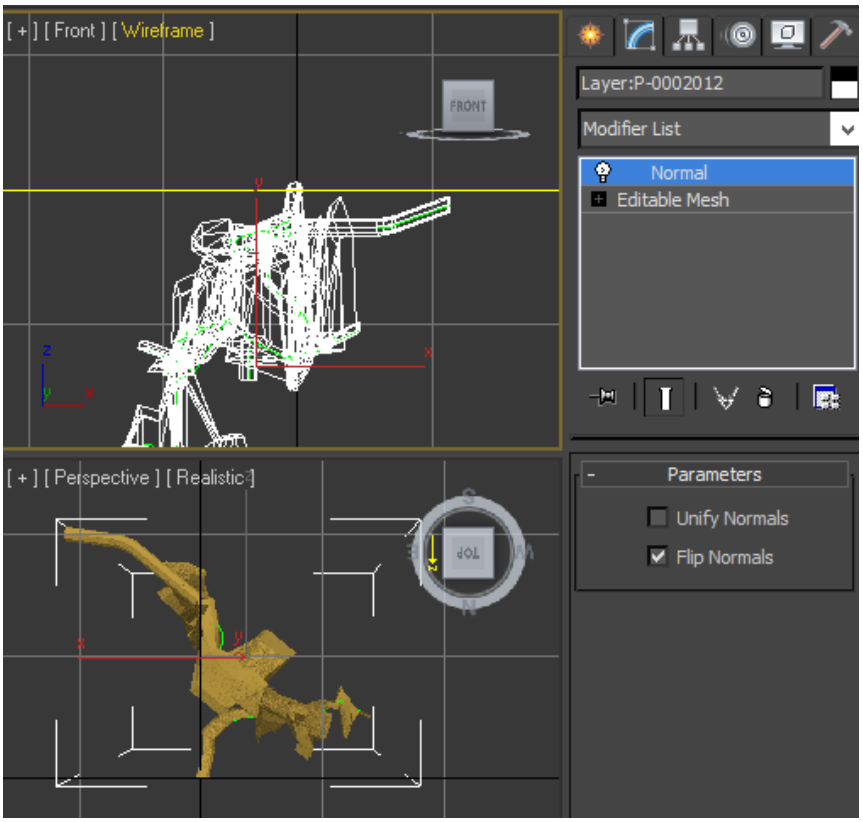

Рисунок 1. Пещера в формате DFX

Плюсы импорта пещеры в формате DFX:

- Быстрота создания модели
- Цельная модель: удобно при трансформировании и перемещении

Минусы:

- Модель не пригодна для точечного редактирования
- Нарушены нормали (отображается задняя стенка, а не передняя)
- Не отображаются гроты (грот превращается в закольцованный ход)

Вывод. Способ пригоден только в случае, если необходимо быстро создать примерную модель пещеры без дальнейшего редактирования. Мне показалось, что этого мало: потребовалось навести лоск у модели. Проведя анализ готовых решений, пришел к выводу: на данный момент

единственный способ – создание вручную. В этой статье хочу показать, как можно автоматизировать один из этапов этого процесса.

Пробежимся немного по основам 3D моделирования. За основу возьмем примитив – цилиндр.

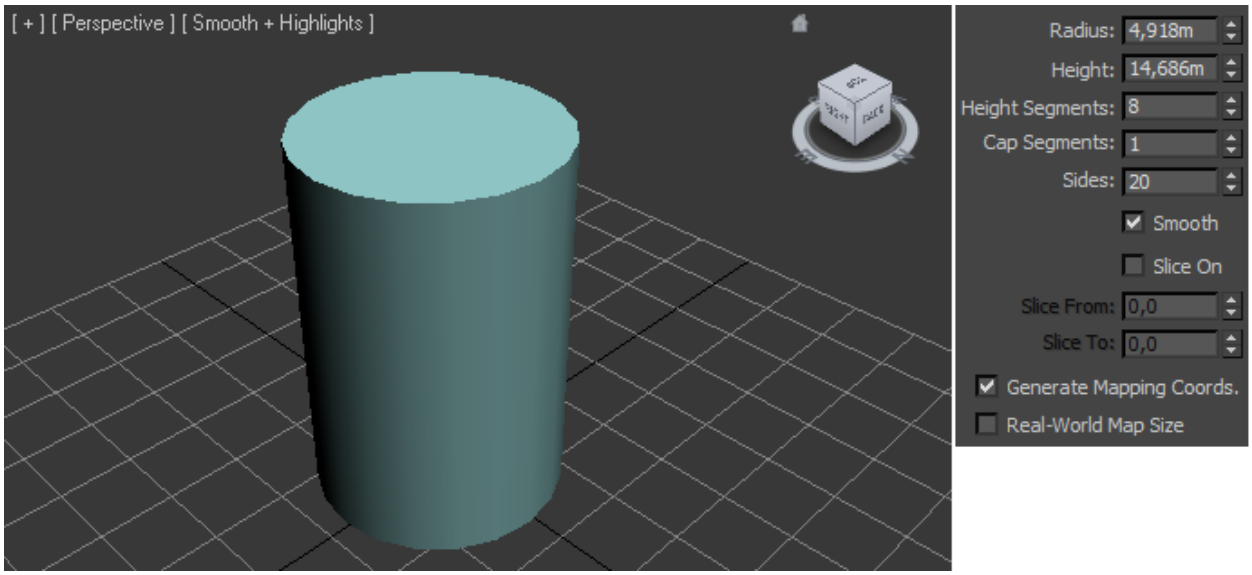

Рисунок 2. Цилиндр

У него есть основные параметры: Радиус, Высота, Вертикальные сегменты, Тыльные сегменты, Количество точек. Изменив несколько параметров (Вертикальные сегменты:2, Количество точек:8) из цилиндра получили фигуру, в основании которой лежит восьмиугольник.

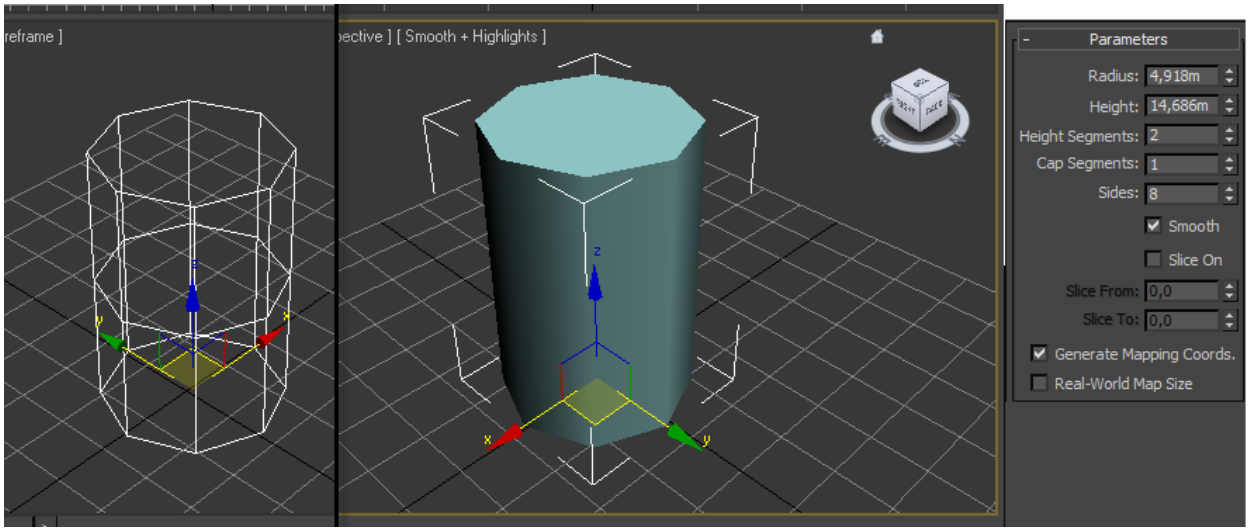

Рисунок 3. Модификация цилиндра

Теперь применим к этой фигуре модификатор трансформации «FFD 2x2x2». Модификатор представляет собой параллелепипед, изменяя вершины которого – деформируется фигура.

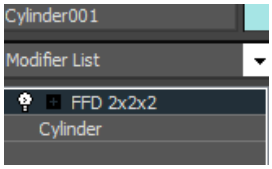

Рисунок 4. Модификатор FFD 2x2x2

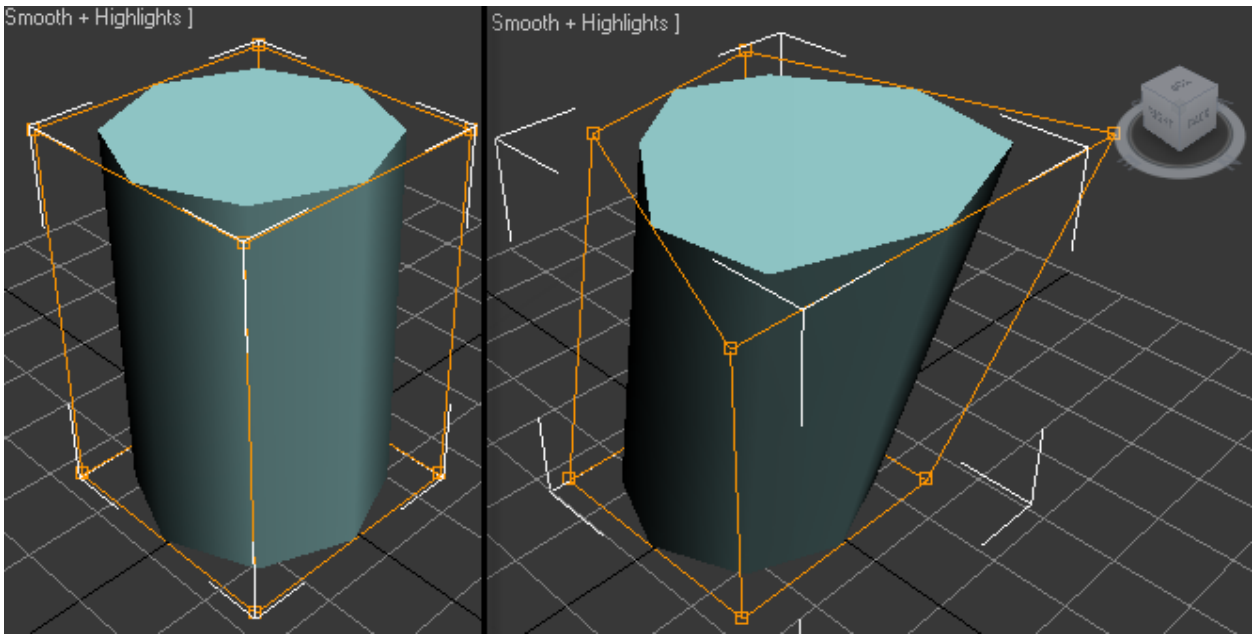

Рисунок 5. МодификаторFFD 2x2x2

Представим, данный цилиндр это наш пикет, у которого можно задать длину, азимут и угол. Используя вершины параллелепипеда, можно указать расстояние до стен, тогда эта фигура становится фрагментом пещеры. Забегая вперед, создавать эти фигуры можно с помощью модуля скриптов, но для этого следует понять геометрическую модель данного процесса.

Координаты пикета. Опорная точка, относительно которой происходит движение и вращение находится в основании нашей фигуры - координата (0;0;0).

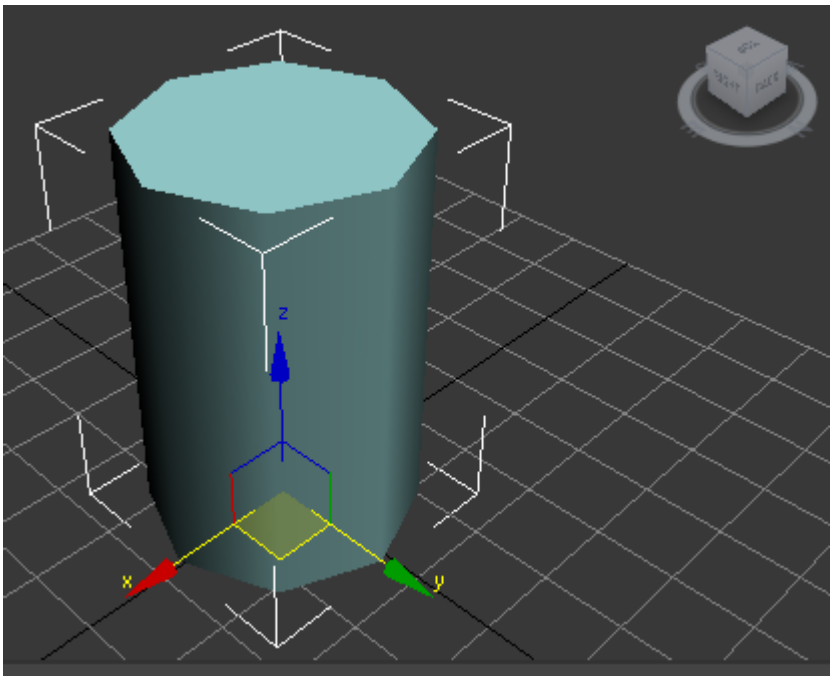

Рисунок 6. Опорная точка

Азимут и угол. Вращение задается углами относительно координатных осей. Азимут по оси OZ, угол – OX. Для соответствия сторонам света необходимо ввести поправки: вращение Z = 180 - азимут; вращение X = 90 – угол.

В таблице 1 представлена пикетажка основной нитки хода с вычисленными приращениями по каждой оси. По формуле:

$$\begin{cases} dx = l \cdot Cos\left(\frac{(az-90)\cdot\pi}{180}\right) \cdot Cos\left(\frac{ug\cdot\pi}{180}\right) \\ dy = -l \cdot Cos\left(\frac{(az-90)\cdot\pi}{180}\right) \cdot Sin\left(\frac{ug\cdot\pi}{180}\right) \\ dz = l \cdot Sin\left(\frac{ug\cdot\pi}{180}\right) \end{cases}$$

, где:
l - Длина;
az - Азимут;
ug - Угол;

Таблица 1

От	До	Length	Azim	Inc	dx	dy	dz	X	Y	Z
0	1	2,5	178	-41	0,07	-1,89	-1,64	0	0	0
1	2	2,6	200	-34	-0,74	-2,03	-1,45	0,07	-1,89	-1,64
2	4	2,6	244	-28	-2,06	-1,01	-1,22	-0,67	-3,91	-3,09
4	5	1,7	261	2	-1,68	-0,27	0,06	-2,73	-4,92	-4,31
5	6	2,9	167	14	0,63	-2,74	0,70	-4,41	-5,18	-4,26

Зная приращения, можно рассчитать координаты каждого пикета. Используя инструмент «скрипт», создадим нитку пещеры.

В начале скрипта определим функцию создающую вектор

```
fn setvectorpxpypzplenpazim  pug =
(
  pradius = 0.1; --Радиу слинии
  cyl = cylinder Radius:pradius Height:plen HeightSegs:2 Sides:8; --Создание вектора

  cyl.rotation.z_rotation = 180 - pazim; --Поворот азимута
  cyl.rotation.x_rotation = 90-pug; -- Угол
  cyl.pos = [px,py,pz] --Смещение
)
```

Таблица 2

№	X	Y	Z	Length	Azim	Inc
1	0.0	0.0	0.0	25.0	178.0	-41.0
2	0.7	-18.9	-16.4	26.0	200.0	-34.0
3	-6.7	-39.1	-30.9	26.0	244.0	-28.0
4	-27.3	-49.2	-43.1	17.0	261.0	2.0
5	-44.1	-51.8	-42.6	29.0	167.0	14.0

В таблице 2 преобразованы данные для обработки скриптом (изменен десятичный разделить, длины увеличены в 10 раз).

Допишем скрипт, используя функцию Setvector:

Setvector	0.0	0.0	0.0	25.0	178.0	-41.0
Setvector	0.7	-18.9	-16.4	26.0	200.0	-34.0
Setvector	-6.7	-39.1	-30.9	26.0	244.0	-28.0
Setvector	-27.3	-49.2	-43.1	17.0	261.0	2.0
Setvector	-44.1	-51.8	-42.6	29.0	167.0	14.0

Весь скрипт находится в файле setvector.ms (см.Приложение).

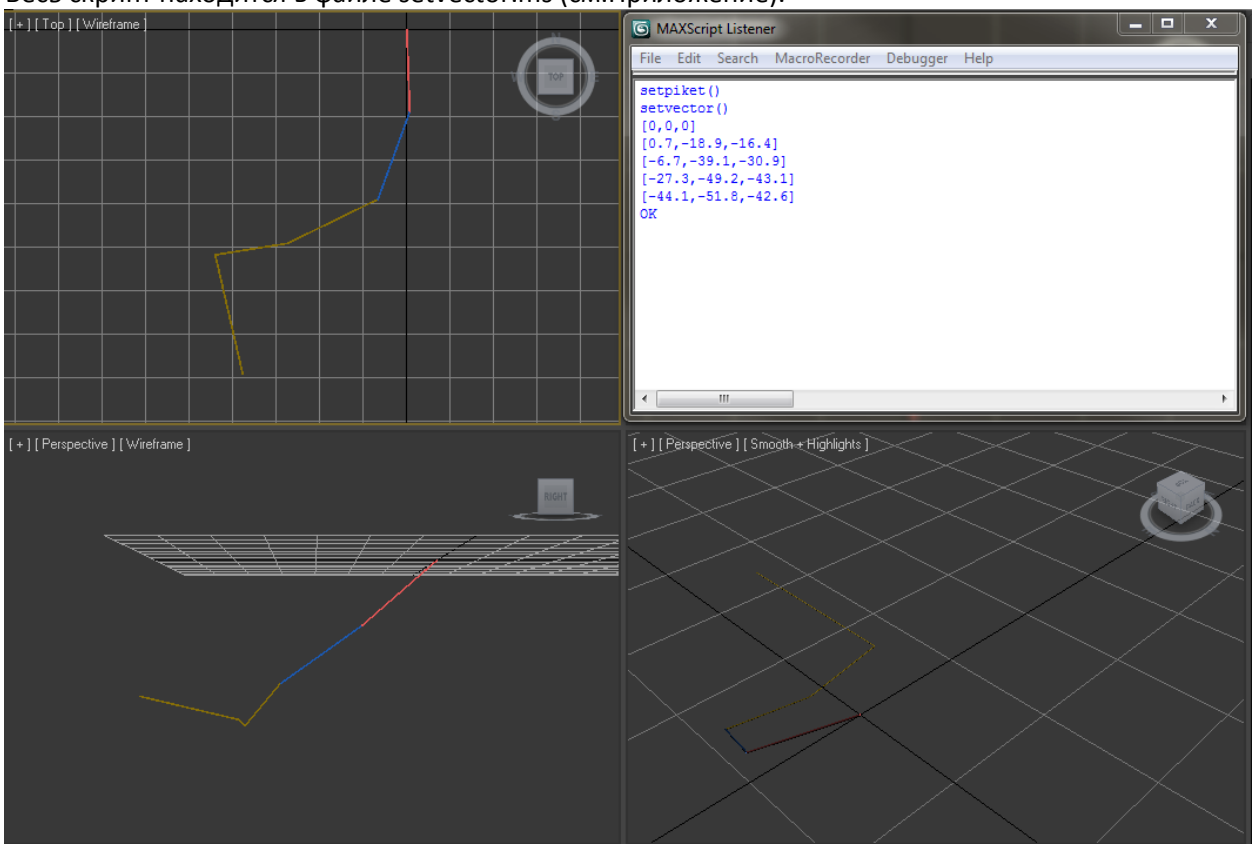

Рисунок 7. Результат

Самая сложная часть это вычисление координат, в этом примере специально упрощена пикетажка, на практике нитка может расходиться, возможна съемка пикетов в обратную сторону. Способы формирования таблицы - это отдельная тема, которая в этой статье рассматриваться не будет.

Формирование стен. При топосъемке расстояния до стен (LRUD) снимаются только в начале пикета, но для нашего сегмента необходимо знать расстояния до стен на конце вектора. Один из способов – брать значение у примыкающего пикета. В таблице 3 L1R1U1D1 – родные значение, L2R2U2D2 – значенияLRUDу соседа.

Таблица 3

От	До	X	Y	Z	Length	Azim	Inc	L1	R1	U1	D1	L2	R2	U2	D2
0	1	0.0	0.0	0.0	25.0	178.0	-41.0	4.0	7.0	5.0	0.0	0.0	10.0	7.0	10.0
1	2	0.7	-18.9	-16.4	26.0	200.0	-34.0	0.0	10.0	7.0	10.0	10.0	4.0	7.0	14.0
2	4	-6.7	-39.1	-30.9	26.0	244.0	-28.0	10.0	4.0	7.0	14.0	10.0	0.0	12.0	18.0
4	5	-27.3	-49.2	-43.1	17.0	261.0	2.0	10.0	0.0	12.0	18.0	17.0	4.0	13.0	14.0
5	6	-44.1	-51.8	-42.6	29.0	167.0	14.0	17.0	4.0	13.0	14.0	17.0	4.0	13.0	14.0

Модификатор «FFD 2x2x2». Прежде чем приступить к написанию скрипта, необходимо понять, как работает модификатор деформации «FFD 2x2x2». Как говорилось ранее: у нас имеется параллелепипед, вершины которого управляющие точки деформации. Нечетные точки находятся снизу, четные – сверху.

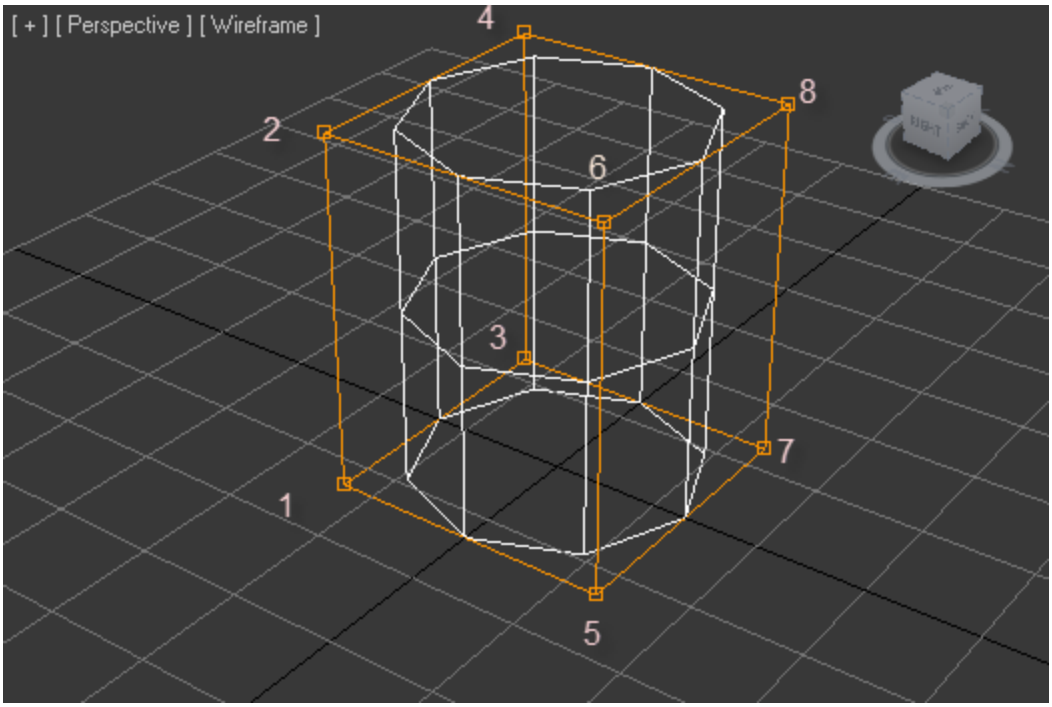

Рисунок 8. Управляющие точки

Рассмотрим нижнюю плоскость (1,3,5,7). Не зависимо от размеров координаты точек:

$$1: (0; 0; 0)$$
$$3: (0; 1; 0)$$
$$5: (1; 0; 0)$$
$$7: (1; 1; 0)$$

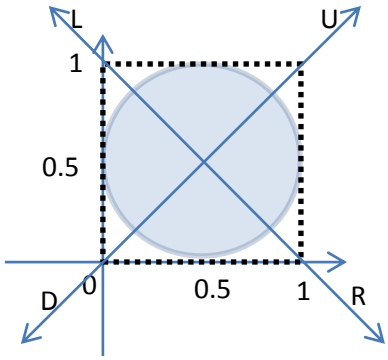

Рисунок 9.

Из «Рисунка 9» видно, что центр фигуры находится в точке (0.5;0.5). Изменяя положение вершин четырехугольника по направлениям векторов LRUD можно задать расстояния до стен. В данном случае направления LRUD не соответствуют реальности, необходимо развернуть фигуру на 45° относительно точки (0.5;0.5).

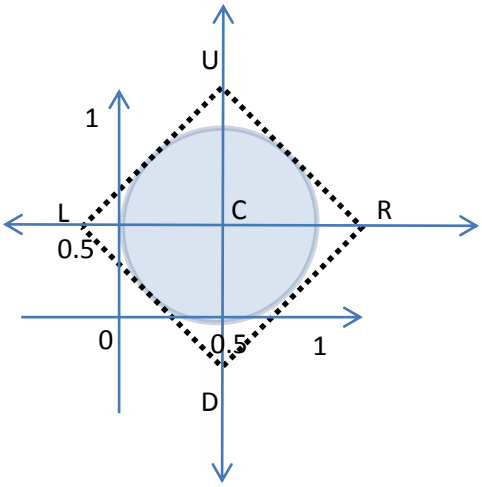

Рисунок 10.

Наша задача - сопоставить смещение LRUD относительно плоскости (1;0;1). Диаметр окружности (цилиндра) логично взять по правилу:

$$d_{cyl} = \max(L + R, D + U)$$

, где:

d_{cyl} – Диаметр цилиндра

LRUD – Расстояния до стен

max – функция, возвращает максимально из двух значений

Получим аналогичное значение для плоскости модификатора.

Рассмотрим треугольник DCR, из «Рисунка 9» видно, что сторона DR = 1, а так как фигура LRUD представляет собой квадрат, то треугольник DCR – равнобедренный, следовательно, угол ∠CDR = 45°.

$$r_m = CD = \frac{\sqrt{2}}{2}$$

$$d_m = UD = 2 \cdot r_m = 2 \cdot \frac{\sqrt{2}}{2} = \sqrt{2}$$

Сопоставим системы координат:

$$d_m : d_{cyl}$$
$$d_p : D$$

$$d_p = \frac{d_m \cdot D}{d_{cyl}} = \frac{\sqrt{2} \cdot D}{d_{cyl}}$$

Так как UD ||OY, то координата точки смещения

$$(0.5; 0.5 - d_p)$$

Фигура LRUD представляет собой квадрат, остальные смещения рассчитываются по такой же формуле

$$d_p = \frac{\sqrt{2} \cdot D}{d_{cyl}}$$
$$u_p = \frac{\sqrt{2} \cdot U}{d_{cyl}}$$
$$l_p = \frac{\sqrt{2} \cdot L}{d_{cyl}}$$
$$r_p = \frac{\sqrt{2} \cdot R}{d_{cyl}}$$

Новые координаты:

$$D\colon\ (0.5; 0.5\ -\ d_p)$$
$$U\colon (0.5; 0.5\ +\ u_p)$$
$$L\colon (0.5\ +\ l_p; 0.5)$$
$$R\colon (0.5\ -\ r_p; 0.5)$$

Реализация метода в Setpiket.ms (см.Приложение).

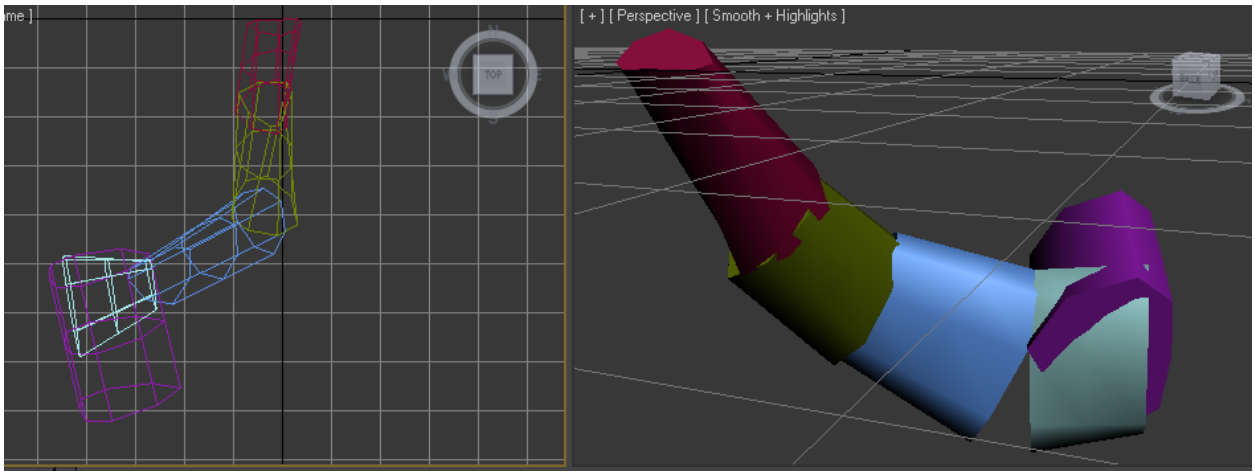

Рисунок 11. Результат

Фрагменты готовы, используя редактор точек (модификатора «EditMesh»), устраняем пробелы.

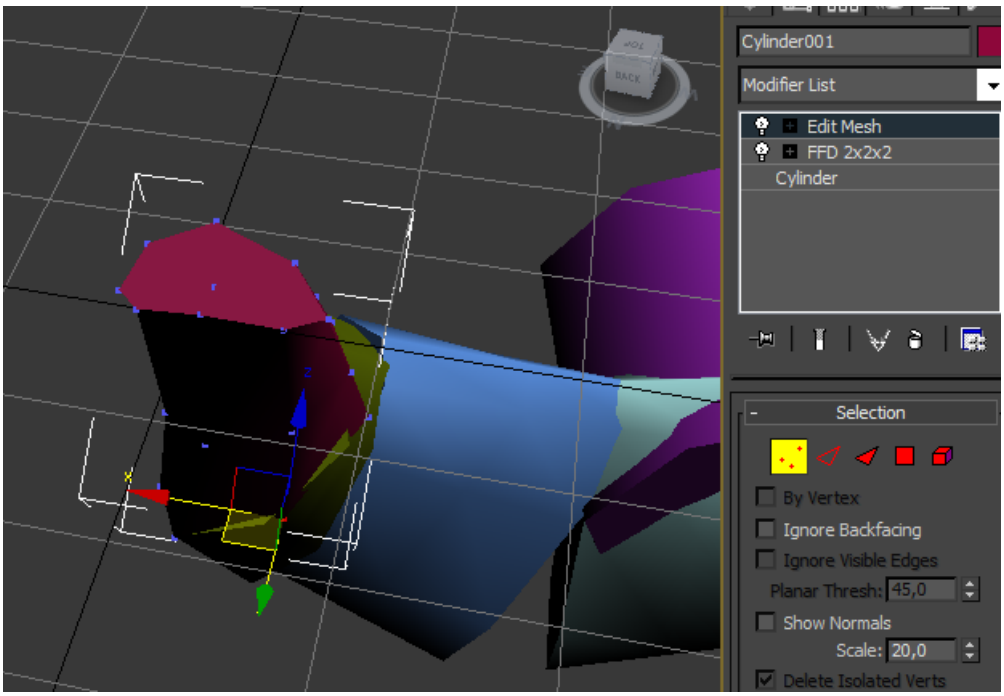

Рисунок 12. Редактирование точек

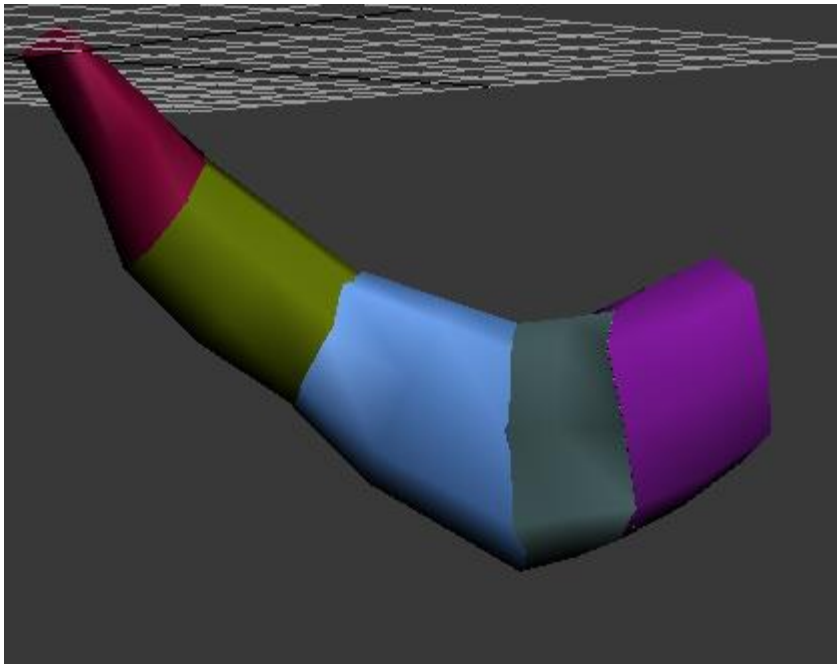

Рисунок 13. Результат

В данной статье был рассмотрен метод автоматизации создания фрагментов пещеры для последующей ее обработки. Недостаток данной методики - необходимость вычисления дополнительных полей в журнале пикетажки. Требуется разработать программный продукт, решающий эту задачу. На выходе мы имеем, удобную для редактирования, модель пещеры.

Приложение

Листинг 1. setvector.ms

```
fn setvector px py pz plen  pazim  pug =
(
	pradius = 0.1;--Radius of line
	cyl = cylinder Radius:pradius Height:plen HeightSegs:2 Sides:8; --create vector

	cyl.rotation.z_rotation = 180 - pazim; --Azimut
	cyl.rotation.x_rotation = 90-pug; -- Clino
	cyl.pos = [px,py,pz] --Move
)

setvector	0.0	0.0	0.0	25.0	178.0	-41.0
setvector	0.7	-18.9	-16.4	26.0	200.0	-34.0
setvector	-6.7	-39.1	-30.9	26.0	244.0	-28.0
setvector	-27.3	-49.2	-43.1	17.0	261.0	2.0
setvector	-44.1	-51.8	-42.6	29.0	167.0	14.0
```

Листинг 2. Setpiket.ms

```
fn setpiket px py pz plen  pazim  pug  pleft1 pright1 pup1 pdown1  pleft2 pright2 pup2 pdown2 =
(

	pradius1 = pdown1 + pup1;
	if (pleft1 + pright1 > pdown1 + pup1) then pradius1 = pleft1 + pright1;
	pradius2 = pdown2 + pup2;
	if (pleft2 + pright2 > pdown2 + pup2) then pradius2 = pleft2 + pright2;
	pradius = pradius2;
	if (pradius1 > pradius2) then pradius = pradius1;

	pradius = pradius / 2.0;

	cyl = cylinder Radius:pradius Height:plen HeightSegs:2 Sides:8;

	cyl.rotation.z_rotation = 180 - pazim;
	cyl.rotation.x_rotation = 90-pug;
	cyl.pos = [px,py,pz]

	addModifier cyl (ffd_2x2x2());
	animateAll cyl.ffd_2x2x2;

	ad1 = (sqrt(2) * pdown1) / (2 * pradius);
	au1 = (sqrt(2) *  pup1) / (2 * pradius);
	al1 = (sqrt(2) *  pleft1) / (2 * pradius);
	ar1 = (sqrt(2) *  pright1) / (2 * pradius);

	ad2 = (sqrt(2) * pdown2) / (2 * pradius);
	au2 = (sqrt(2) *  pup2) / (2 * pradius);
	al2 = (sqrt(2) *  pleft2) / (2 * pradius);
	ar2 = (sqrt(2) *  pright2)		/ (2 * pradius);

	--down
	cyl.ffd_2x2x2.control_point_1 = [0.5, 0.5 - ad1, 0];
```

```
        cyl.ffd_2x2x2.control_point_2 = [0.5, 0.5 - ad2, 1];
        --up
        cyl.ffd_2x2x2.control_point_7 = [0.5, 0.5 + au1, 0];
        cyl.ffd_2x2x2.control_point_8 = [0.5, 0.5 + au2, 1];
        --right
        cyl.ffd_2x2x2.control_point_3 = [0.5 - ar1, 0.5 , 0];
        cyl.ffd_2x2x2.control_point_4 = [0.5 - ar2, 0.5 , 1];
        --left
        cyl.ffd_2x2x2.control_point_5 = [0.5 + al1, 0.5 , 0];
        cyl.ffd_2x2x2.control_point_6 = [0.5 + al2, 0.5 , 1];

        addModifier cyl (edit_mesh());
)


setpiket 0.0 0.0 0.0 25   178     -41     4     7     5     0     0     10    7     10
setpiket 0.7 -18.9 -16.4 26 200  -34     0     10    7     10    10    4     7     14
setpiket -6.7 -39.1 -30.9 26 244 -28     10    4     7     14    10    0     12    18
setpiket -27.3 -49.2 -43.1 17 261 2      10    0     12    18    17    4     13    14
setpiket -44.1 -51.8 -42.6 29 167 14     17    4     13    14    17    4     13    14
```